Response Time Upper Bounds for Fixed Priority Real-Time Systems

Robert Davis and Alan Burns
Real-Time Systems Research Group
University of York

Outline

- **Background and motivation**
  - Why are we interested in response time upper bounds?
- **Recap on standard analysis**
  - System model and Response Time Analysis
- **Response time upper bound**
  - Derivation
  - Application to pre-emptive, co-operative, and non pre-emptive scheduling problems
- **Empirical investigations**
  - Comparison with other simple schedulability tests
- **Summary and conclusions**

Background

- **Fixed priority scheduling**
  - Widely used in real-time embedded systems:
    - electronic control units and communications networks in automobiles, digital set-top boxes, medical systems, space systems, and mobile phones.
  - Supported by nearly all commercial RTOS
  - Supported by schedulability analysis
    - Response Time Analysis exists for system models with broad scope
      - blocking, release jitter, arbitrary deadlines etc.
      - co-operative and non-pre-emptive scheduling
    - Exact analysis has pseudo-polynomial complexity
    - Can almost always be used to determine schedulability of industrial scale systems in reasonable time, despite theoretical complexity results

Motivation

- **Why are we interested in Response Time Upper Bounds?**
  - Improve practical efficiency of exact schedulability test
    - Check on a task-by-task basis if schedulable according to upper bound
    - Only compute exact response time for a task when upper bound > deadline
    - Typical tasksets, majority of tasks are easily schedulable, so using an upper bound can result in **significant improvements in efficiency**

    [R.I. Davis, A. Zabos, and A. Burns, "Efficient Exact Schedulability Tests for Fixed Priority Pre-emptive Systems" *IEEE Transactions on Computers* September 2008 (Vol. 57, No. 9) pp. 1261-1276]

Motivation

- **Other uses of Response Time Upper Bounds?**
  - Can be used when complexity / execution time of exact response time analysis is a limitation
  - Interactive system design tools
    - Sensitivity analysis requires results of large numbers of schedulability test be available in HCI timescales
  - System optimisation via search
    - Using simulated annealing / GAs with schedulability as a cost function
  - Dynamic systems
    - Online admission of new tasks / applications with stringent start-up constraints

System Model

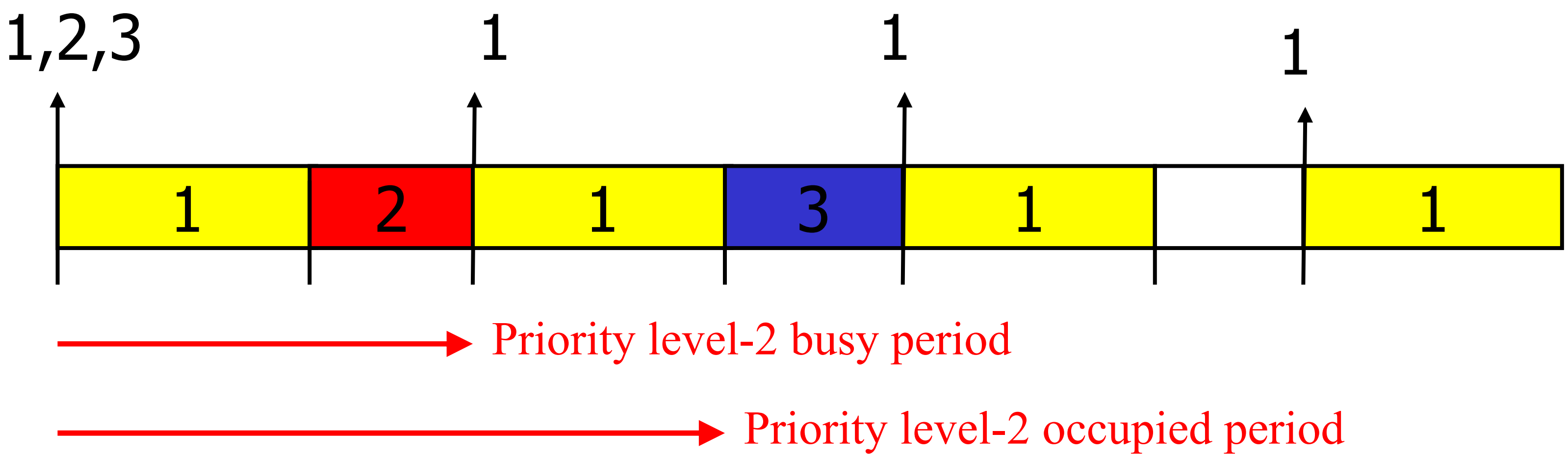
- **Single processor**
  - Static set of *n* tasks τ_i
  - Fixed Priority Scheduling
- **Task parameters**
  - Worst-case execution time C_i
  - Sporadic/periodic arrivals: minimum inter-arrival time T_i
  - Arbitrary Deadlines $D_i \le T_i$, $D_i > T_i$
  - Blocking factor B_i
  - Release jitter J_i, from arrival to release
  - Worst-case response time R_i, from release to completion
- **Independent arrival times**
  - Potential for simultaneous release

System Model

- **Task scheduling**
  - Pre-emptive
  - Co-operative / Non-pre-emptive
    - Final non-pre-emptive section $F_i \leq C_i$
- **Blocking**
  - Access to mutually exclusive shared resources according to the Stack Resource Policy (SRP) – [Baker 1991]
  - Blocking factor B_i
    - Longest time a lower priority task can execute at priority i or higher due to SRP or non-pre-emptive sections

Terminology

- **Priority *i* busy period**
  - Time interval during which the processor is busy executing at priority i or higher until it completes some computation C at priority i
- **Priority *i* occupied period**
  - Time interval during which the processor is busy executing at priority i or higher until it has completed some computation C at priority i and is available to continue executing computation at priority i



Response time analysis: recap

- **Pre-emptive scheduling**
  - General model, arbitrary deadlines, release jitter, blocking etc.
  - Determine length of **multiple busy periods** starting at a critical instant, extending to completion of qth invocation of task τ_i

$$w_i^{n+1}(q) = B_i + (q+1)C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{w_i^n(q) + J_j}{T_j} \right\rceil C_j$$

  - Response time given by $R_i(q) = w_i^{n+1}(q) - qT_i$
  - Start with $w_i^0(q) = B_i + (q+1)C_i$
  - Iterate until $w_i^{n+1}(q) = w_i^n(q)$ or $w_i^{n+1}(q) - qT_i > D_i - J_i$
  - Worst-case response time $R_i = \max_{\forall q}(w_i(q) - qT_i)$
    - Check values of q until an invocation completes before the next release
  - Schedulable if $R_i \le D_i - J_i$

Response time analysis: recap

- **Non-pre-emptive scheduling**
  - Determine length of **multiple occupied periods** starting at critical instant, extending to time at which the qth invocation can start its final non-pre-emptable section

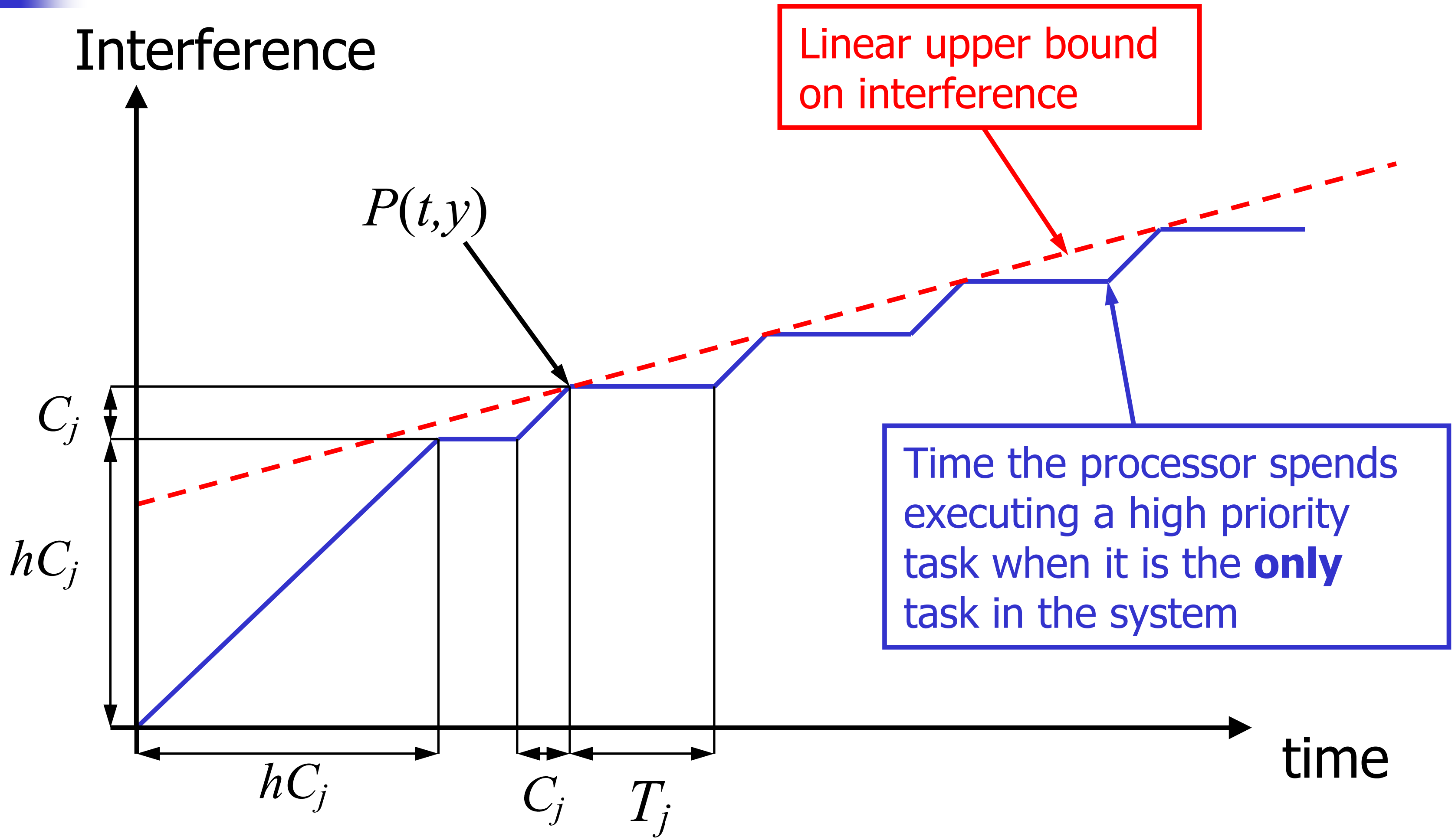
$$v_i^{n+1}(q) = B_i + (q+1)C_i - F_i + \sum_{\forall j \in hp(i)} \left(\left\lfloor \frac{v_i^n(q) + J_j}{T_j} \right\rfloor + 1 \right) C_j$$

  - Response time given by $R_i(q) = v_i^{n+1}(q) + F_i - qT_i$
  - Start with $v_i^0(q) = B_i + (q+1)C_i - F_i$
  - Iterate until $v_i^{n+1}(q) = v_i^n(q)$ or $v_i^{n+1}(q) + F_i - qT_i > D_i - J_i$
  - Worst-case response time $R_i = \max_{q=0,1..Q_i-1}(v_i(q) + F_i - qT_i)$
    - Number of invocations to check related to number of invocations Q in the busy period for pre-emptive scheduling
  - Schedulable if $R_i \leq D_i - J_i$

Derivation of the Upper Bound

- **Approach**
  - Method introduced by Bini & Baruah 2007
  - Idea is to derive an **upper bound on interference** from each high priority task assuming that it is the **only task** in the system
  - Use these upper bounds on interference to determine an upper bound on task response time
- **Extended here to**
  - Account for blocking and release jitter
  - Cater for co-operative and non-pre-emptive scheduling (as well as the pre-emptive case)

Interference Upper Bound



Interference Upper Bound

- **Determine number of invocations h that execute consecutively from time $t = 0$**
  - Number of invocations released at $t = 0$ is $\lfloor J_j / T_j \rfloor + 1$
  - Subsequent releases at times

    $$\left(\lfloor J_j / T_j \rfloor + 1\right) T_j - J_j + (k-1) T_j$$

    for $k = 1,2,3\ldots$
  - Number of subsequent releases within the interval of consecutive execution is given by the largest k :

    $$\left(\lfloor J_j / T_j \rfloor + 1\right) C_j + (k-1) C_j \geq \left(\lfloor J_j / T_j \rfloor + 1\right) T_j - J_j + (k-1) T_j$$

    $$k = \lfloor J_j / (T_j - C_j) \rfloor - \lfloor J_j / T_j \rfloor$$
  - Hence:

    $$h = \lfloor J_j / (T_j - C_j) \rfloor + 1$$

Interference Upper Bound

- **Point** $P(t,y)$

$$y = hC_j + C_j = \left(\left\lfloor J_j/(T_j - C_j)\right\rfloor + 1\right)C_j + C_j$$

$$t = hT_j - J_j + C_j = \left(\left\lfloor J_j/(T_j - C_j)\right\rfloor + 1\right)T_j - J_j + C_j$$

- **Interference upper bound:**

$$I_j^{UB}(t) = U_j t + U_j J_j + C_j(1 - U_j)$$

  - For all higher priority tasks:

$$\sum_{\forall j \in hp(i)} I_j^{UB}(t) = t \sum_{\forall j \in hp(i)} U_j + \sum_{\forall j \in hp(i)} (U_j J_j + C_j(1 - U_j))$$

Busy Period Upper Bound

- **Busy Period Upper Bound on time for processor to complete C execution at priority i**
  - Intersection of the lines:

$$y = t$$

$$y = C + t \sum_{\forall j \in hp(i)} U_j + \sum_{\forall j \in hp(i)} (U_j J_j + C_j (1 - U_j))$$

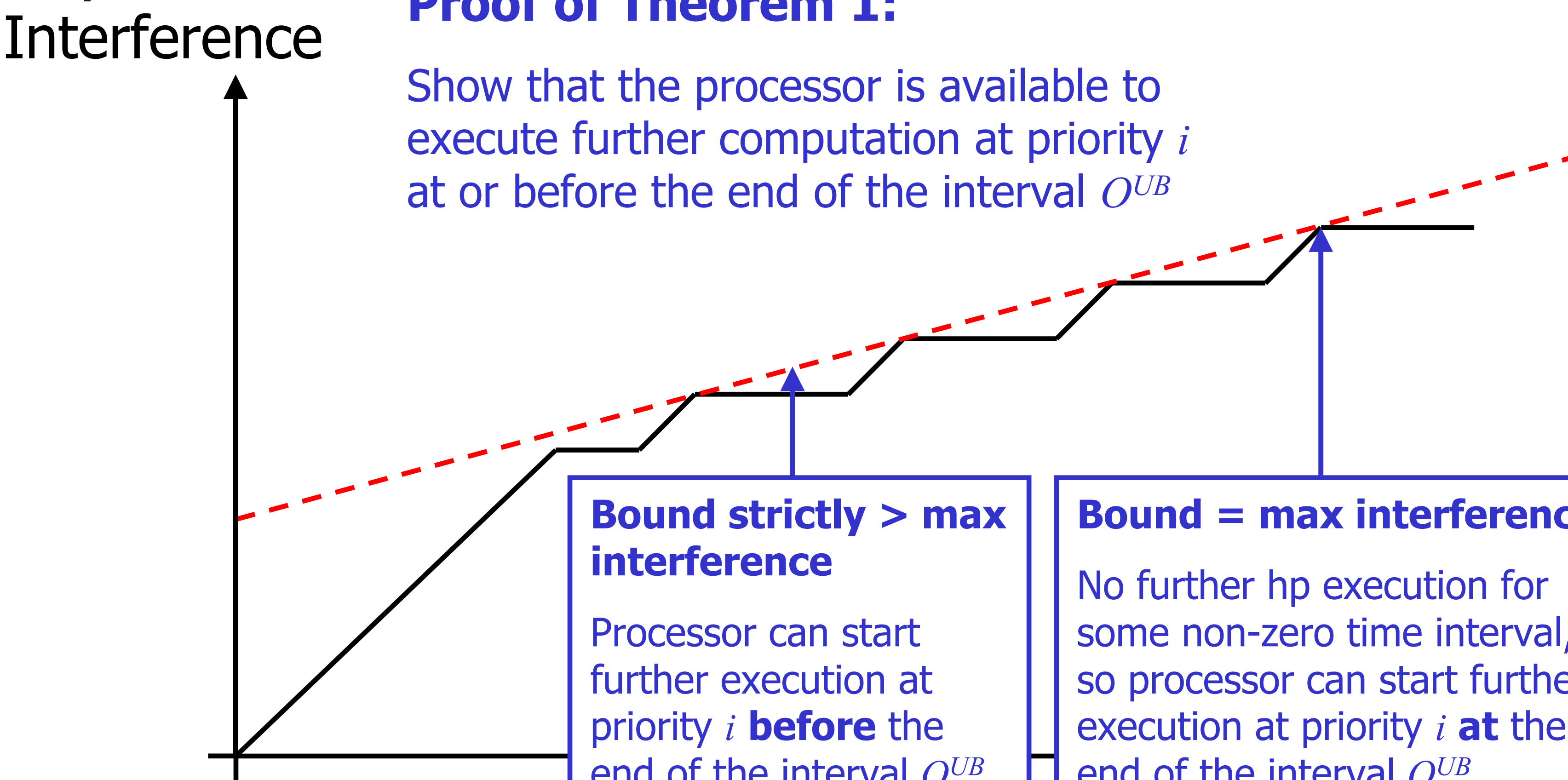
$$O_i^{UB}(C) = \frac{C + \sum_{\forall j \in hp(i)} (U_j J_j + C_j (1 - U_j))}{1 - \sum_{\forall j \in hp(i)} U_j}$$

- Theorem 1: $O_i^{UB}(C)$ is also an upper bound on the **occupied period** for computation C at priority i

Occupied Period Upper Bound

Proof of Theorem 1:

Show that the processor is available to execute further computation at priority i at or before the end of the interval O^{UB}



Bound strictly > max interference

Processor can start further execution at priority i **before** the end of the interval O^{UB}

Bound = max interference

No further hp execution for some non-zero time interval, so processor can start further execution at priority i **at** the end of the interval O^{UB}

Response Time Upper Bound

- **Pre-emptive case**
  - Occupied period upper bounds the pre-emptive busy period

$$W_i^{UB}(q) = \frac{B_i + (q+1)C_i + \sum_{\forall j \in hp(i)} (U_j J_j + C_j(1-U_j))}{1 - \sum_{\forall j \in hp(i)} U_j}$$

  - Response time bound for each invocation $R_i^{UB}(q) = W_i^{UB}(q) - qT_i$
  - Comparing response time bounds for different invocations

$$R_i^{UB}(q) - R_i^{UB}(q+1) = T_i - \frac{C_i}{1 - \sum_{\forall j \in hp(i)} U_j} \geq 0$$

  - Worst-case response time upper bound (first invocation)

$$R_i^{UB} = \frac{B_i + C_i + \sum_{\forall j \in hp(i)} (U_j J_j + C_j(1-U_j))}{1 - \sum_{\forall j \in hp(i)} U_j}$$

Response Time Upper Bound

- **Co-operative (and non-pre-emptive) case**
  - Upper bound on occupied time

$$V_i^{UB}(q) = \frac{B_i + (q+1)C_i - F_i + \sum_{\forall j \in hp(i)} (U_j J_j + C_j (1 - U_j))}{1 - \sum_{\forall j \in hp(i)} U_j}$$

  - Bound for each invocation $R_i^{UB}(q) = V_i^{UB}(q) + F_i - qT_i$
  - Comparing response times for different invocations:

$$R_i^{UB}(q) - R_i^{UB}(q+1) = T_i - \frac{C_i}{1 - \sum_{\forall j \in hp(i)} U_j} \geq 0$$

  - Worst-case response time upper bound (first invocation)

$$R_i^{UB} = \frac{B_i + C_i - F_i + \sum_{\forall j \in hp(i)} (U_j J_j + C_j (1 - U_j))}{1 - \sum_{\forall j \in hp(i)} U_j} + F_i$$

Linear time sufficient test

- **Closed form Response Time Upper bound**

$$R_i^{UB} = \frac{B_i + C_i - F_i + \sum_{\forall j \in hp(i)} (U_j J_j + C_j (1 - U_j))}{1 - \sum_{\forall j \in hp(i)} U_j} + F_i$$

$$\forall i \quad R_i^{UB} \le D_i - J_i$$

- **Widely applicable to processor and network scheduling**
  - Arbitrary deadlines, blocking, release jitter
  - Task scheduling
    - Pre-emptive: $F_i = 0,$
    - Co-operative: $0 < F_i < C_i$
    - Non-pre-emptive $F_i = C_i$
- **Via incremental summation, highest priority first, can determine schedulability of n tasks in O(n) time**

Response Time Upper Bound

- **Example taskset**

	C_i	T_i	D_i	J_i	B_i	$D_i - J_i$	R_i	R_i^{UB}
τ_1	3	10	10	2	0	8	3	3
τ_2	15	100	50	5	10	45	37	40
τ_3	15	200	200	5	10	195	58	75
τ_4	40	400	400	50	20	350	153	191
τ_5	30	1000	500	50	50	450	282	404
τ_6	200	1000	1000	100	0	900	682	876

Empirical investigation

- **Compares Response Time Upper bound with**
  - Exact response time analysis
  - Sufficient tests
    - Utilisation based test (Liu & Layland 1973)
    - RBound (Lauzac et al. & Buttazzo 2003)
    - Hyperbolic bound (Bini et al. 2003)
  - Sufficient tests adapted to cater for arbitrary deadlines, blocking, and release jitter

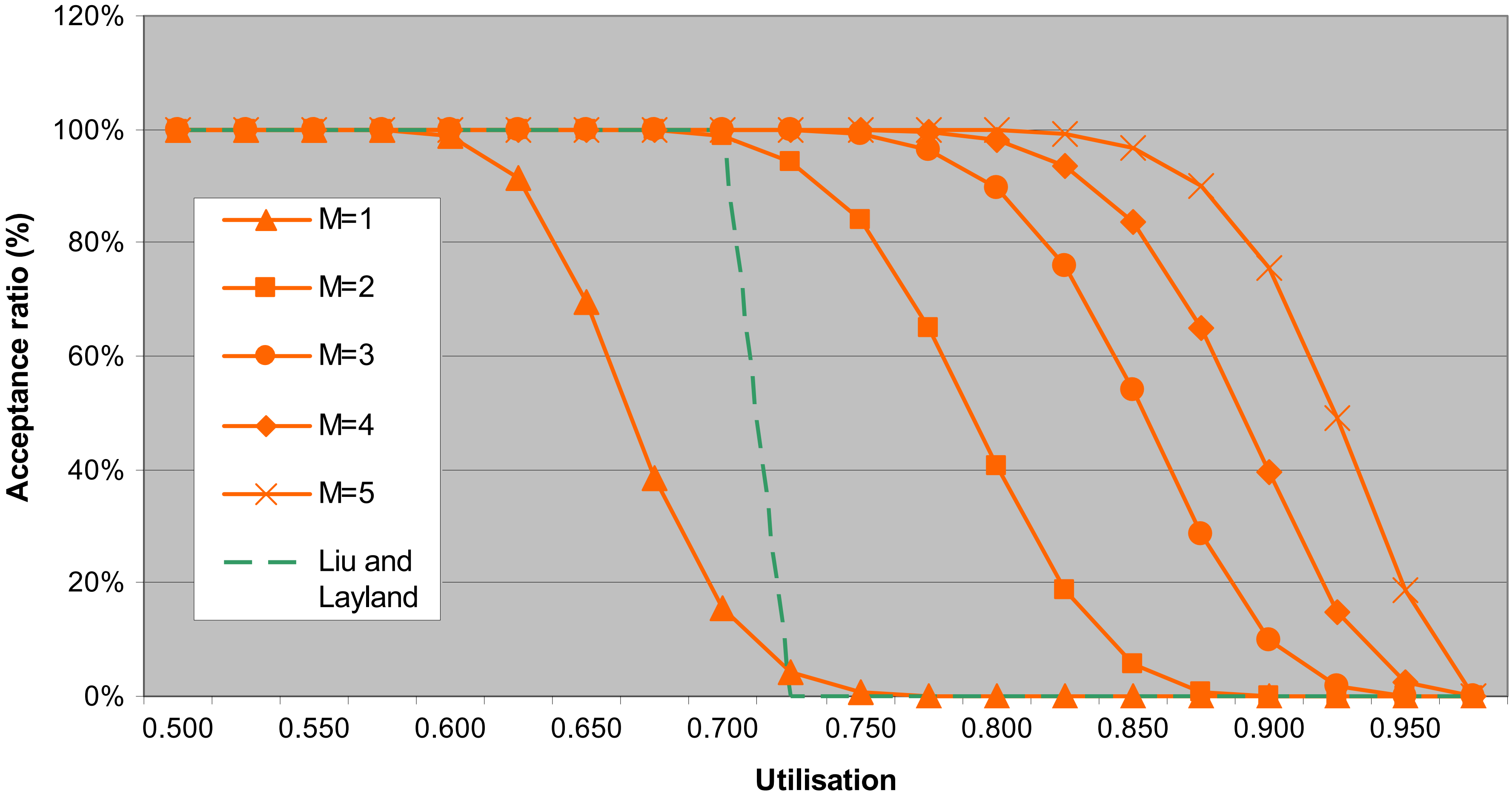
$$\frac{C_i + B_i}{D_i - J_i} + \sum_{j=1..i-1} \frac{C_j}{D_j - J_j} \leq i(2^{1/i} - 1)$$

Experiments

- **Varied:**
  - Number M of orders of magnitude ranges used for task period selection (1-5, default = 2)
    - E.g. for $M=3$ task periods chosen from 3 ranges [100-1000, 1000-10,000, 10,000-100,000]
  - Utilisation (5% – 95%, default 60%)
  - Deadlines (0.05 – 0.95 of period, default = period,)
  - Blocking factors (0.5 – 9.5 of execution time, default =0)
  - Release jitter (0.05 – 0.95 of period, default =0)

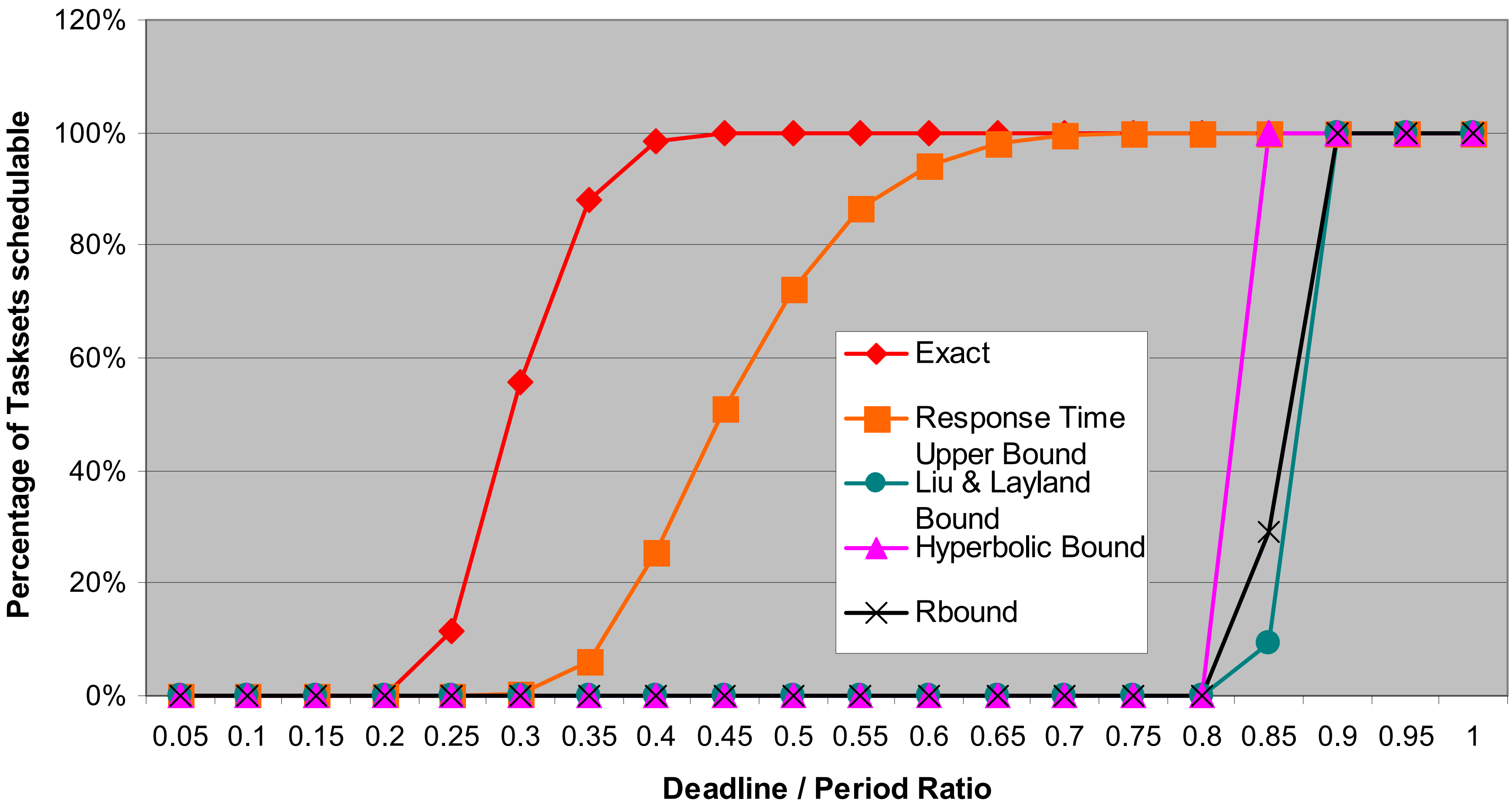
  - 10,000 tasksets for each x-axis point on graphs
  - Taskset cardinality = 24

Expt 1: Range of task periods



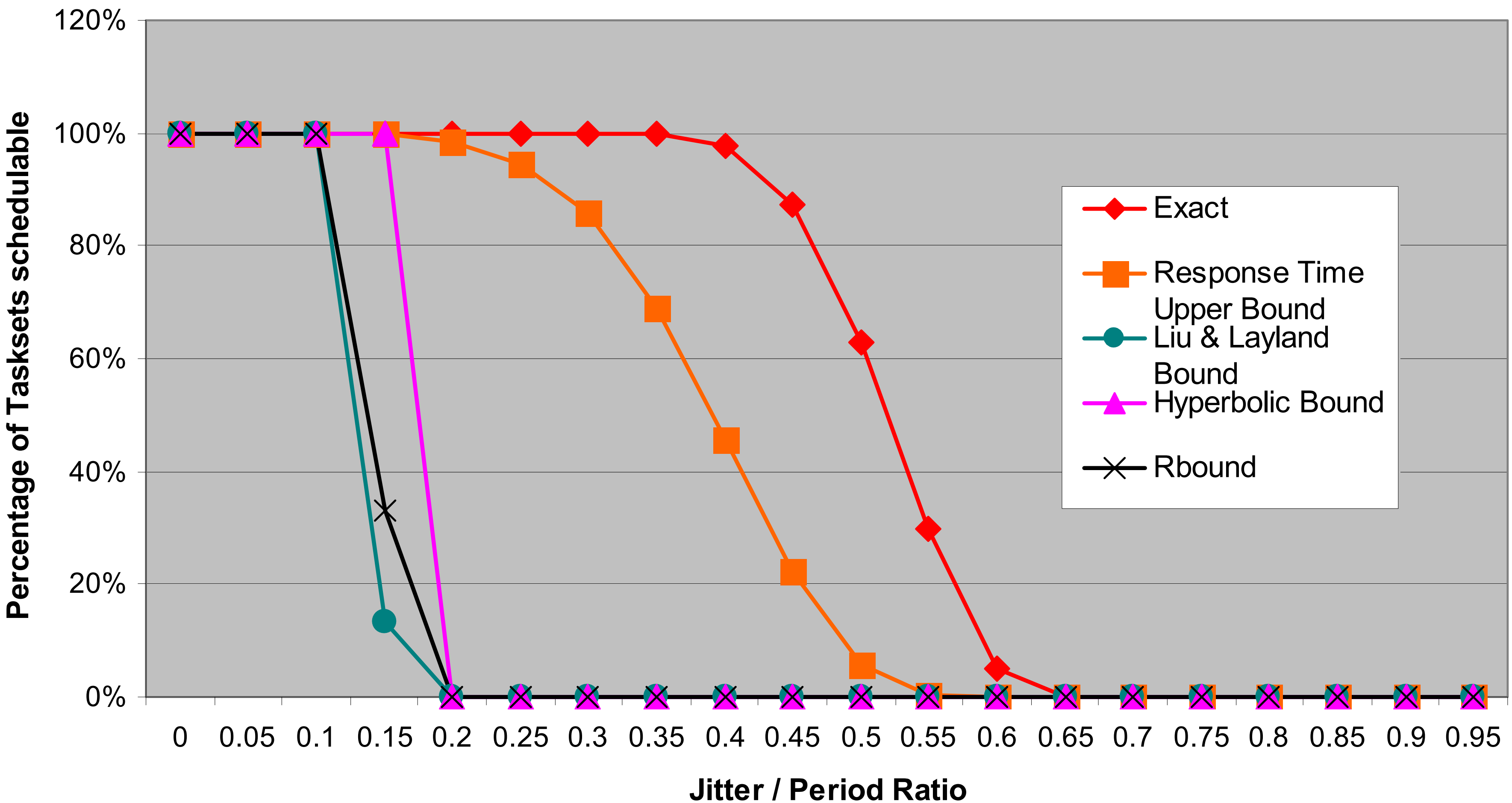
(Fixed parameters: $D = T$, $B = 0$, $J = 0$)

Expt 2: Deadline : period ratio



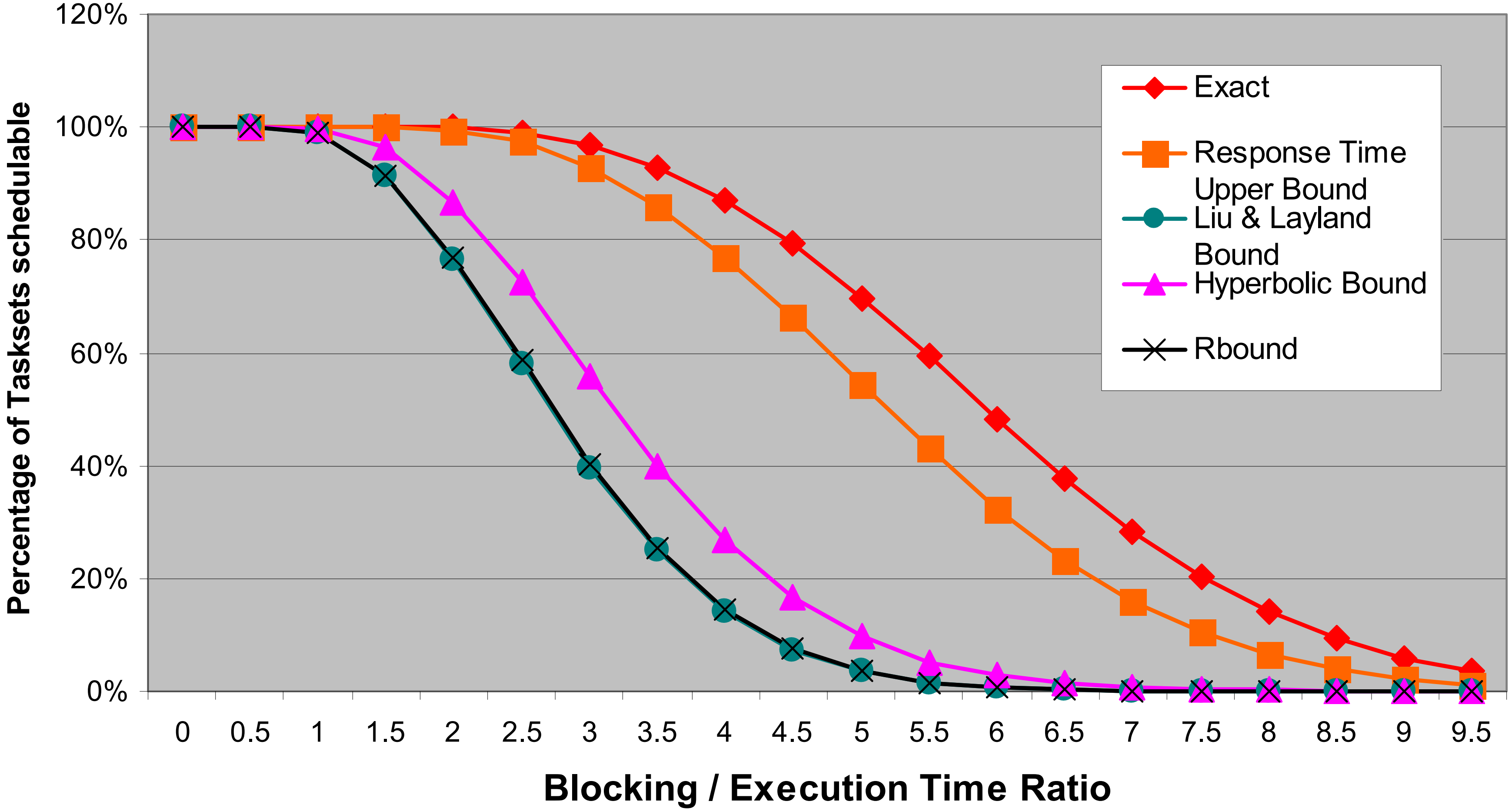
(Fixed parameters: *M* = 2, *U* = *60%*, *B* = 0, *J* = 0)

Expt 3: Jitter : period ratio



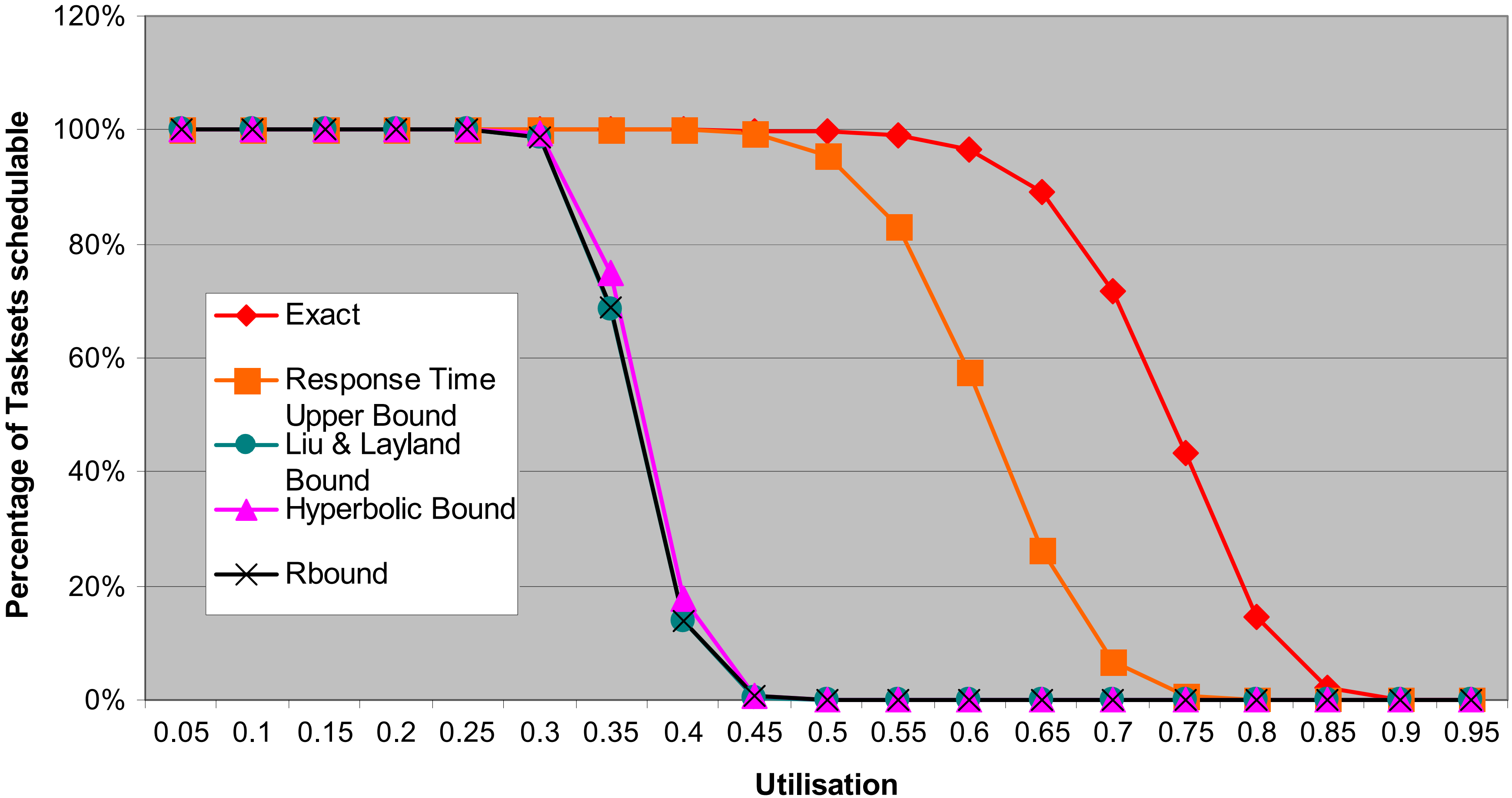
(Fixed parameters: $M = 2$, $U = 60\%$, $D = T$, $B = 0$)

Expt 4: Blocking : ET ratio



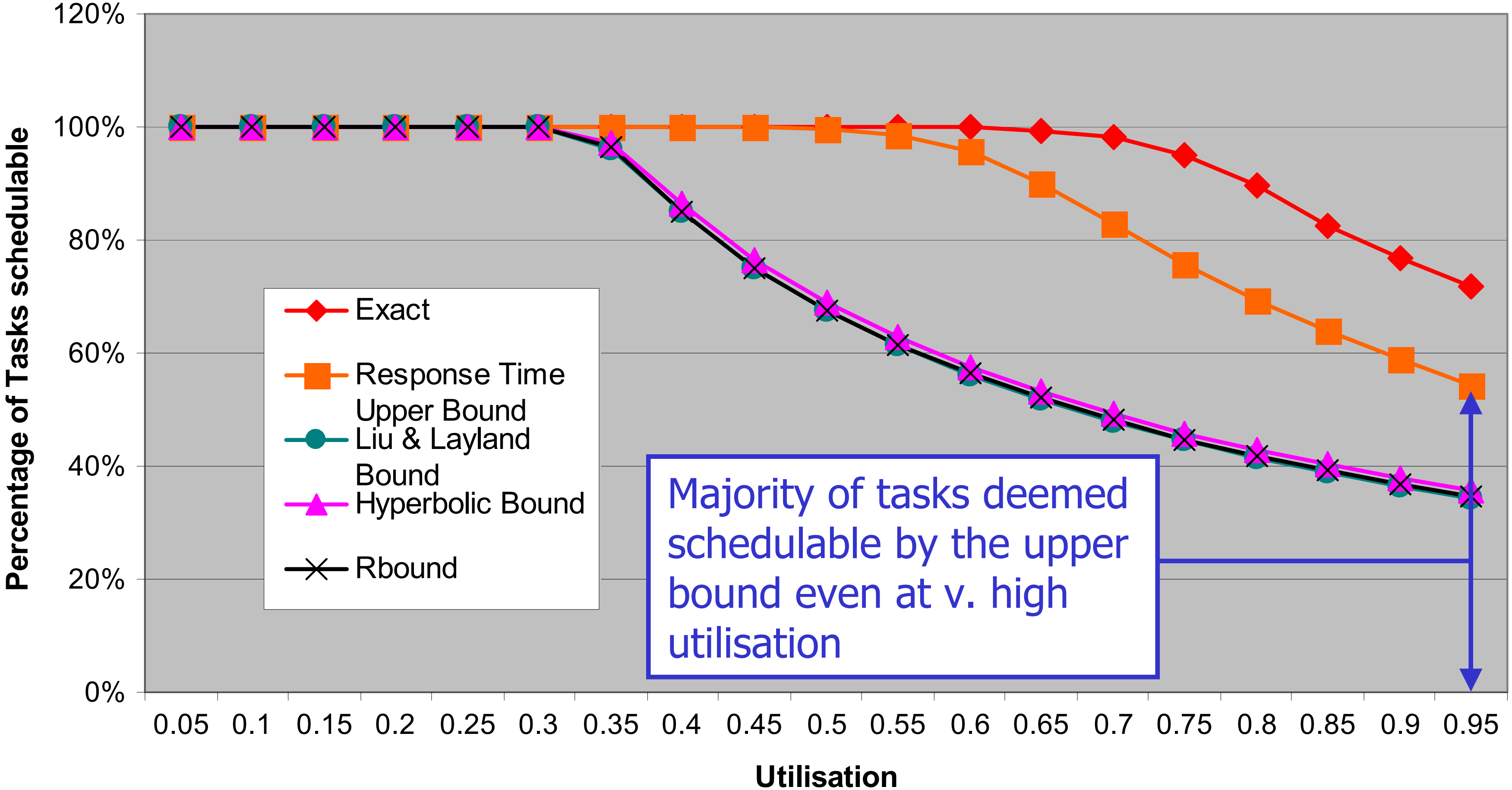
(Fixed parameters: $M = 2$, $U = 60\%$, $D = T$, $J = 0$)

Expt 5: All parameters varied



(Fixed parameters: $M = 2$, Varied parameters: $D = 0.5T$ to $1.0T$, $J = 0.5D$ to $1.0D$, $B = 0$ to $1.0C$)

Expt 6: Tasks schedulable



(Parameters as Expt. 1)

Summary and conclusions

- **Derived a response time upper bound**
  - Based on the idea of a linear bound on interference
  - Extended scope to a general system model supporting
    - Blocking, release jitter (and arbitrary deadlines)
  - Shown that the bound can be applied to pre-emptive, co-operative, and non-pre-emptive scheduling
- **Single closed form upper bound applicable to a wide range of real-time systems and networks**
  - Forms a linear time sufficient schedulability test
    - $O(n)$ time for n tasks
  - Can be used to significantly improve the efficiency of exact response time analysis in practical applications
    - Used on a task-by-task basis; only perform exact calculation when sufficient test fails

Summary and conclusions

- **Other uses of the Response Time Upper Bound**
  - Online admission tests
    - With stringent time constraints on start-up
  - Interactive system design tools
    - Response Time Upper Bound is continuous and differentiable w.r.t. parameters
    - No nasty surprises: small increase / decrease in a parameter cannot cause a sudden large increase in the response time upper bound
  - System optimisation via search (future research)
    - Early stage of search; find region of interest in search space using continuous upper bounds
    - Use exact analysis to find solution

Questions?

$$R_i^{UB} = \frac{B_i + C_i - F_i + \sum_{\forall j \in hp(i)} (U_j J_j + C_j (1 - U_j))}{1 - \sum_{\forall j \in hp(i)} U_j} + F_i$$

The End